(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平7－282095

(43)公開日　平成７年(1995)10月27日

(51)Int.Cl.[6]	識別記号	庁内整理番号	Ｆ Ｉ	技術表示箇所
Ｇ０６Ｆ　17/50				
		7623－5L	Ｇ０６Ｆ　15/ 60	３６０　Ｋ

審査請求　未請求　請求項の数３　ＯＬ　（全 16 頁）

(21)出願番号　特願平6－70383

(22)出願日　平成６年(1994)４月８日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中1015番地
(72)発明者　藤澤　久典
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(74)代理人　弁理士　長谷川　文廣　（外２名）

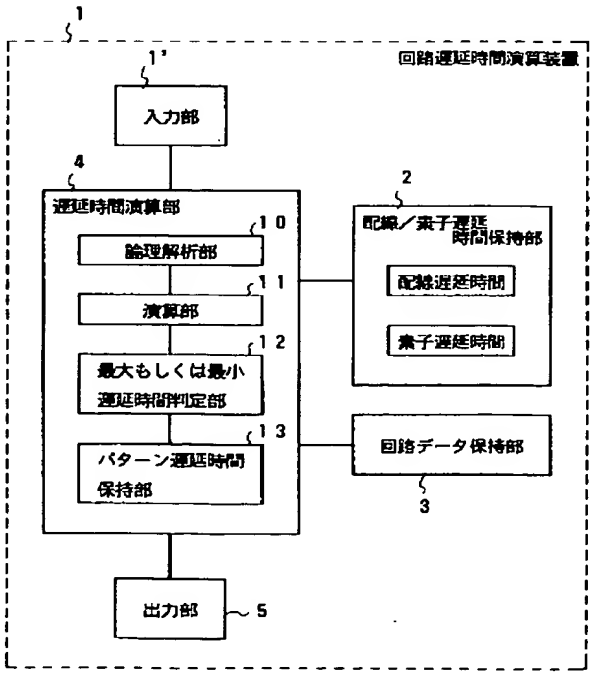
(54)【発明の名称】　回路遅延時間演算装置

(57)【要約】

【目的】　組合せ回路の遅延時間を演算する遅延時間演算装置に関し，高速にかつ正確に最大もしくは最小遅延時間を算出することを目的とする。

【構成】　回路データを入力する入力部(1')と，配線および素子の遅延時間を保持する配線／素子遅延時間保持部(2)と，回路データを保持する回路データ保持部(3)と，回路の遅延時間を演算する遅延時間演算部(4)と，回路の遅延時間の演算結果を出力する出力部(5)とを備え，遅延時間演算部(4)は回路に入力される外部入力パターンの論理が回路の各ノードに伝播する論理を解析する論理解析部(10)を備え，論理解析部(10)は，ノードの論理を与える外部入力パターンの有無を判定し，ノードの論理を実現する外部入力パターンが存在する論理のみにより最大遅延時間を算出する構成をもつ。

本発明の基本構成



1

【特許請求の範囲】

【請求項１】　回路データを入力する入力部(1')と，配線および素子の遅延時間を保持する配線／素子遅延時間保持部(2) と，回路データを保持する回路データ保持部(3) と，回路の遅延時間を演算する遅延時間演算部(4) と，回路の遅延時間の演算結果を出力する出力部(5) とを備え，該遅延時間演算部(4) は回路に入力される外部入力パターンの論理が回路の各ノードに伝播する論理を解析する論理解析部(10)を備え，該論理解析部(10)は，ノードの論理を与える外部入力パターンの有無を判定し，該ノードの論理を実現する外部入力パターンが存在する論理のみにより最大遅延時間を算出することを特徴とする回路遅延時間演算装置。

【請求項２】　請求項１において，該論理解析部(10)は，ノード毎に該ノードの論理を与える外部入力パターンの論理を求める論理式もしくは論理情報を保持し，素子の出力論理を与える素子の入力側のノードの論理を実現する外部入力パターンの有無を該素子の入力側のノードの論理式もしくは論理情報に基づいて判定することを特徴とする回路遅延時間演算装置。

【請求項３】　回路データを入力する入力部(1')と，配線および素子の遅延時間を保持する配線／素子遅延時間保持部(2) と，回路データを保持する回路データ保持部(3) と，回路の遅延時間を演算する遅延時間演算部(4) と，回路の遅延時間の演算結果を出力する出力部(5) とを備え，該遅延時間演算部(4) は，二分決定グラフ法により素子の入力端子および配線等のノードの論理を実現する外部入力パターンを求める論理解析部(10)を備え，該二分決定グラフにより各ノードのグラフ求め，求めたグラフの葉にその葉の論理を伝播する遅延時間を付加し，対象ノードと該対象ノードの前段ノードとの論理関係および前段のノードの遅延時間に基づいて対象ノードのグラフと遅延時間を求めることを特徴とする回路遅延時間演算装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は，組合せ回路の遅延時間を演算する回路遅延時間演算装置に関する。コンピュータを利用して回路設計する場合，回路の入出力間の遅延時間を求める必要がある。回路が複雑な場合にはこのような遅延時間の計算は膨大な計算量を必要とするので，効率的な演算装置が望まれる。

【０００２】

【従来の技術】組合せ回路の入出力の遅延時間を算出する従来の方法は，次の通りである。まず，各ゲートの入出力端子間の最大（もしくは最小）遅延時間とゲートの最大（もしくは最小）遅延時間を求めておく（遅延時間は最大遅延時間と，最小遅延時間を求める場合の２通りがあるが，以下，最大遅延時間について説明する。最小遅延時間についても方法は同じである）。

2

【０００３】図１８を参照して従来の遅延時間算出方法を説明する。図１８において，１１０はアンドゲートである。

【０００４】１１１はインバータである。１１２は前段回路である。a_1，a_3 は外部入力端子である。

【０００５】a_2，a_4 はゲート入力端子であって，アンドゲート１１０の入力端子である。a_5 はゲート出力端子であって，アンドゲート１１０の出力端子である。

【０００６】a_6 はゲート入力端子であって，インバータの入力端子である。a_7 はゲート出力端子であって，インバータ１１１の出力端子である。a_8 は外部出力端子である。

【０００７】外部入力端子a_1，a_3 と外部出力端子a_8 間の遅延時間を算出する。アンドゲート１１０のa_2，a_5 間およびa_4，a_5 間およびインバータ１１１のa_6，a_7 間の遅延時間はあらかじめ求めておく。また，配線a_1 a_2，配線a_3 a_4，配線a_5 a_6，配線a_7 a_8 の遅延時間もあらかじめ求めておく。

【０００８】まず，各外部入力端子a_1，a_3 から対象ゲート１１０，１１１のゲート入力端子までの最大遅延時間にそのゲート入力端子からそのゲート出力端子までの遅延時間を加算したもののうち最大のものをその対象ゲートのゲート出力端子までの最大遅延時間とする。

【０００９】例えば図１８の場合，a_1 a_2 間の遅延時間にa_2 a_5 の遅延時間を加算した遅延時間と，a_3 a_4 間の遅延時間にa_4 a_5 の遅延時間を加算した遅延時間のうち大きい方をa_5 の最大遅延時間t_5 とする。

【００１０】次に，求めたゲート出力端子a_5 の最大遅延時間t_5 に，そのゲート出力端子a_5 の接続先のゲート入力端子a_6 までの遅延時間を加算したものをそのゲート入力端子a_6 の最大遅延時間t_6 とする。

【００１１】さらに，インバータ１１１に対して同様の手順で遅延時間を求め，外部入力端子a_1，a_3 から外部出力端子a_8 までの最大遅延時間を求める。

【００１２】

【発明が解決しようとする課題】従来の回路の遅延時間算出方法は，ゲート出力端子の遅延時間を算出する場合，対象ゲートのゲート入力端子の論理パターン（ゲート入力パターン）を全ての論理の組み合わせに基づいて求めていた。

【００１３】例えば，図１８の回路の場合に，外部入力端子a_1，a_3 の前に別の回路（例えばＥＯＲ）の出力は（０，０）か（１，１）でしかないものとする。このとき，アンドゲート１１０での遅延時間は，ゲート入力パターンが（０，０），（１，１）である場合のみを考慮して最大遅延時間を算出すれば良いが，従来は，それ以外のあらゆる論理の組み合わせのパターンに基づいて最大遅延時間を求めていた。

【００１４】そのため，有り得ないゲート入力パターンに対しても遅延時間を計算することとなり，実際以上に

最大遅延時間を過大に見積もったり，最小遅延時間を過小に見積もったりすることがあった。また，演算速度を低下させてもいた。

【００１５】本発明は，高速にかつ正確に最大もしくは最小遅延時間を算出することのできる回路遅延時間演算装置を提供することを目的とする。

【００１６】

【課題を解決するための手段】本発明は，各ノード（外部入力端子，ゲート入力端子，ゲート出力端子，外部出力端子，配線端子等）の論理値を実現する対象入力端子の論理パターンを記憶し，直前に接続しているノードの遅延時間からそのノードの遅延時間を計算するにあたっては，実際に実現するパターンのみを考慮して計算するようにした。

【００１７】前段のノードから，対象ノードの遅延時間を計算する方法は次の通りである。ノードがゲート出力端子の場合，まずゲート入力端子の論理値の組み合わせに対して，その組合せが実現するような外部入力パターン（外部入力端子の入力パターン）が存在するかどうかチェックする。もし存在すればその外部入力パターンの組み合わせに対するゲート出力端子の論理および遅延時間を計算する。もし，要求している遅延時間が最大遅延時間であれば，当ゲートの論理を実現するゲート入力端子の各ゲート入力パターンの最大遅延の中から，最大遅延のものをその論理の最大遅延とする。また，要求している遅延時間が最小遅延であれば，当ゲートの論理を実現するゲート入力端子の各パターンの最小遅延のものをその論理の最小遅延とする。

【００１８】一方，ノードがゲート入力端子の場合には，当ノードに繋がっているゲート出力端子の遅延に，配線遅延を加えたものを，当ノードの遅延時間とする。配線による伝播では論理は変化しないので各論理を実現する外部入力パターンの組に変化はない。

【００１９】また，本発明は，このように遅延時間を求めるために必要なノード論理の判定を論理式もしくは論理情報の記憶による場合の他に，二分決定グラフ法により必要な論理のみを求める場合の両方で実現した。

【００２０】図１の本発明の基本構成の説明に先立ち，二分決定グラフ法による場合について説明する。図２(a)は二分決定グラフ法の説明図である（ナンドゲートの場合）。

【００２１】図２（a)に示すように，二分決定グラフ（以下，グラフと称する）は節（○）と枝（節（○）と葉（正方形）を結ぶ線），葉（正方形）からなっている。節は入力変数に対応しており，二本の枝を持っている。二本の枝は各節の入力変数の論理値に対応しており，別の節または葉に繋がっている。入力変数の間には優先順位が付けられており，もとの節（Ａ）の入力変数の方が枝を介して接続している節（Ｂ）の入力変数より優先順位が高いものとする。また，葉は，考慮しているノードの論理値の最大または最小遅延情報を持っている。二分決定グラフ法はルートと呼ぶ節をもっており，ルートから葉までの経路（優先順位の低い変数（優先順位の低い節の変数）から高い変数（優先順位の高い節の変数）の節へ行くような経路は考慮しない）は，その経路の節が経路上の論理をとった場合に，到達した葉に対応した論理値を持つことを表している。枝に沿って記述してある数字は，その枝をもつ節の入力変数の論理を表している。正方形（葉）の中は出力論理で，その下の括弧で括られた数値は遅延時間を示している。入力変数の優先順位はＡの方が高い。

【００２２】前段のノードのグラフから，当ノードのグラフを作成する手法は以下の通りである。ノードがゲート出力端子の場合，まずゲート入力端子のグラフに対してゲートの論理演算を行うことにより，当ノードの論理値を表現するグラフを作成する（作成方法は（「“Ｇｒａｐｈ－Ｂａｓｅｄ Ａｌｇｏｒｉｔｈｍｓ ｆｏｒＢｏｏｌｅａｎ Ｆｕｎｃｔｉｏｎ Ｍａｎｉｐｕｌａｔｉｏｎ”，Ｒｎｄａｌ Ｅ．Ｂｒｙａｎｔ，ＩＥＥＥ Ｔｒａｎｓａｃｔｉｏｎ ｏｎ Ｃｏｍｐｕｔｅｒｓ．Ｖｏｌ．Ｃ－３５，Ｎｏ８，ｐｐ６７７～６９１，１９８６，８」参照）。グラフ同士の演算により新しいグラフを作成する過程において，新しく得られるグラフの葉の遅延時間については，その葉が得られる（その葉の論理を実現する）ゲートの入力端子の論理の組み合わせパターンが入力した際の最大（最小）遅延時間を，その葉の遅延時間とする。また，葉のリダクション（同じ論理をもつ葉を一つにまとめる（図２（b)参照）場合には，もし要求している遅延時間が最大遅延時間であれば，最大遅延のものをその論理の最大遅延とし，要求している遅延が最小遅延であれば，最小遅延のものをその論理の最小遅延とする。この処理をグラフ演算に付加することにより該当ノードのグラフと遅延時間が求められる。

【００２３】一方ノードがゲートの入力端子の場合には，当ノードに繋がっているゲートの出力端子のグラフの各論理の遅延に配線による遅延を加えたものを，当ノードのグラフとする。

【００２４】図１は本発明の基本構成を示す図である。図１において，１は回路遅延時間演算装置である。

【００２５】２は配線／素子遅延時間保持部であって，配線の遅延時間と素子（ゲート等）の遅延時間を保持するものである。３は回路データ保持部であって，ノードの結合情報であるネットリスト，回路特性を示すパラメータ等の回路データを保持するものである。

【００２６】４は遅延時間演算部であって，回路の遅延時間を演算するものである。５は出力部であって，演算結果を出力するものである。遅延時間演算部４において，１０は論理解析部であって，各ノードにおける論理を解析するものであり，各ノードの論理を与える外部入力パターン（外部入力端子に与えられる入力変数の組）

5

の論理を求めるための論理式もしくは論理情報（真理値表等）をノード毎に求め，外部入力端子の論理を実現する外部入力パターンの有無を判定するものである。あるいは，二分決定グラフを求めるものである。

【００２７】１１は演算部であって，ノードの有効パターン（ノードの論理を実現する外部入力パターンが存在するノードの論理パターン）に基づいて遅延時間を算出するものである。

【００２８】１２は最大もしくは最小遅延時間判定部であって，計算された遅延時間最大もしくは最小遅延時間を求めるものである。１３はパターン遅延時間保持部であって，ノードの論理を与える外部入力パターンを求めるための論理式もしくは論理情報を保持するとともに，最大遅延時間を保持するものである。

【００２９】

【作用】図１の本発明の基本構成の動作を説明する。図１において，例えば，外部出力端子から外部入力端子に向かって各ノードを遡り，論理解析部１０は外部入力端子から近いノードから順番に対象ノードの遅延時間および論理式もしくは論理情報（ノードの論理を実現する外部入力パターンを求めるための論理式もしくは論理情報であって，以後単に論理式もしくは論理情報と称する）を求める。そして，ノードの論理特性により決められるノードの論理を実現する外部入力パターンの有無を判定する。そして，演算部１１は，ノードの遅延時間を求めるのに必要なノードの論理のうち外部入力パターンの存在するノードの論理についてのみ，その論理が伝播する遅延時間を算出する。最大もしくは最小遅延時間判定部１２は求められた遅延時間のうち対象ノードで最大（もしくは最小）のものを求める。パターン遅延時間保持部１３はその対象ノードの論理式もしくは論理情報および最大遅延時間を保持する。

【００３０】以上の処理を外部入力端子に近いノードから外部出力端子まで順番に繰り返し，外部出力端子の最大もしくは最小遅延時間を求め，出力部５に出力する。図３の回路により具体的に説明する。ゲートG_4に着目する。ゲートG_4のゲート入力端子a_{11}，a_{13}（いずれもノード）について考える。外部入力パターン（a，b，c）に対してa_{11}の論理を１とする論理式は（－a）・（－c）である（－は否定を表す。例えば（－b）はbの否定論理である）。即ち，（－a）・（－c）＝１とするａ，ｃの論理はａ＝０かつｃ＝０である。従って，論理式（－a）・（－c）によりa_{11}の論理１もしくは論理０を与える外部入力パターン（a，b，c）の組を求めることができる。同様にa_{13}の論理式は（－b）・cである。

【００３１】一方，ゲートG_4の出力を１とする入力端子a_{11}，a_{13}のとる論理パターンは（０，０）と（１，１）である。そして，（a_{11}，a_{13}）を（０，０）とする外部入力パターン（a，b，c）は，次の論理式

6

（－（（－a）・（－c））・（－（（－b）・c））＝a・b＋a・（－c）＋b・c＝１とするものである。

【００３２】論理解析部１０は，（a，b，c）の組み合せパターンから，そのような有効な外部入力パターンは（１，＊，０）もしくは（＊，１，１）であることを判定する（＊は０，１いずれでも良い）である。そして，G_4の出力を１とする場合の遅延時間の演算はこの二通りのパターンについてのみなされ，ゲートG_4の遅延時間が求められる。また，ゲートG_4の入力（a_{11}，a_{13}）を（１，１）とする論理式は，

（（－a）・（－c））・（（－b）・c）＝１

である。この式は（a，b，c）の全てのパターンに対して恒等的に０であるので，（a_{11}，a_{13}）を（１，１）とする外部入力パターンは存在しない。従って，論理解析部１０は上記の論理式を満たす外部入力パターンは存在しないことを判定し，遅延時間判定部１２はそのようなゲート入力パターンについては遅延時間の計算は行わない（なお，図３の回路の遅延時間の算出方法の詳細については後述する）。さらに，G_4の出力を０とする場合についても同様に外部入力パターンの有無を判定し，有効なゲート入力パターンについてのみ遅延時間を求める。

【００３３】本発明によれば，素子の入力論理パターンについて，実際に発生する有効なゲート入力パターンについのみ遅延時間を求めるので，少ない演算回数で正確な遅延時間を算出することが可能となる。

【００３４】

【実施例】本発明の実施例の装置構成を説明する前に，図３を参照して本発明の回路遅延時間演算方法について説明する。

【００３５】図３は本発明の実施例を説明するための回路例である。図３において，G_1はゲートであって，インバータである。

【００３６】G_2はゲートであって，ノアゲートである。G_3はゲートであって，ノアゲートである。G_4はゲートであって，排他的論理回路である。

【００３７】A（＝a_1）は外部入力端子（ノード）である。B（＝a_6）は外部入力端子（ノード）である。C（＝a_3）は外部端子（ノード）Aの入力である。

【００３８】a_2，a_4はゲートG_2のゲート入力端子（ノード）である。a_{10}はゲートG_2のゲート出力端子（ノード）である。a_5はゲートG_1のゲート入力端子（ノード）である。

【００３９】a_8はゲートG_1のゲート出力端子（ノード）である。a_7，a_9はゲートG_3のゲート入力端子（ノード）である。a_{12}はゲートG_3のゲート出力端子（ノード）である。

【００４０】a_{11}，a_{13}はゲートG_4のゲート入力端子（ノード）である。a_{14}はゲートG_4のゲート出力端子

（ノード）である。ＯＵＴ（＝a_{15}）は外部出力端子である。

【００４１】説明を簡単にするために，遅延時間の計算において，各ゲート間の配線時間は１とする。また，入力が変化したときに出力がＬからＨに変化する遅延をアップ遅延と呼ぶ。入力が変化したとき，出力がＨからＬに変化する遅延をダウン遅延と呼ぶ。また，インバータG_1のアップ遅延は８，ダウン遅延は６とする。ノアゲートG_2，G_3のアップ遅延は１０，ダウン遅延は７とする。排他的論理回路G_4のアップは９，ダウンは７であるとする。

【００４２】以下ノードa_jのアップ遅延をa_j↑，ダウンをa_j↓で表すものとする。本発明の装置構成を説明する前に，論理式により有効なゲート入力パターン（有効なノードの論理）を求める場合の遅延時間演算方法について説明する（二分決定グラフによらない場合）。

【００４３】外部入力端子Ａ，Ｂ，Ｃから外部出力端子（ＯＵＴ）までの最大遅延時間を計算する。a_{15}から入力端子側にパスを辿る。a_{11}とa_{13}はともに，未処理なので，まずa_{13}を選択する。再び未処理の入力をもつノアゲートG_3にたどりつく。そこで，a_7を選択して，さらに溯り，外部入力ノードa_6に到達する。その遅延時間は０である。a_7の遅延時間はa_6の遅延時間に$a_6 - a_7$間の配線遅延を加えたものである。

【００４４】a_7↑＝a_6↑＋１＝１
a_7↓＝a_6↓＋１＝１
である。

【００４５】また，ノードa_7の論理式は，
ｂ
である。ｂ＝１のときa_7＝１であり，ｂ＝０のときa_7＝０である。a_7のこの論理式もしくは論理条件（真理値表等）をa_7に対応付けて保持する。

【００４６】次は，a_9の遅延時間を計算する。a_9から溯ると，外部入力ノードa_3に到達する。ノードa_3の遅延時間は０であるから，a_5のノードの遅延時間は，a_3の遅延時間に配線遅延１を加えた
a_5↑＝a_3↑＋１＝１
a_5↓＝a_3↓＋１＝１
a_5の論理式は
ｃ
である。ノードa_5にこの論理式もしくは論理情報を保持する。

【００４７】さらに，出力側に戻ってa_8の遅延時間を計算する。a_8はインバータの出力であるから，a_8が１となるのは入力ノードa_5が０の時である。この時，入力パターンはｃ＝０の時であるから，遅延時間は
a_8↑＝a_5↓＋８＝９
である。また，a_8が０となるのは入力ノードa_5が１の時である。この時の入力パターンはｃ＝１の時であるから，遅延時間は，
a_8↓＝a_5↑＋６＝７
となる。また，a_8が１となる時の入力パターンはｃ＝０であるから，論理式は
（－ｃ）
である。ノードa_8にこの論理式もしくは論理情報を保持する。

【００４８】そこで，配線遅延１をアップ遅延とダウン遅延に加えると，
a_9↑＝a_8↑＋１＝１０
a_9↓＝a_8↓＋１＝８
である。

【００４９】a_7，a_9の遅延時間が得られたので，a_{12}の遅延時間を計算する。ゲート入力端子（a_7，a_9）の組合せは（０，０），（１，０），（１，１），（０，１）である。a_9の論理式は（－ｃ），a_7の論理式はｂである。従って，a_{12}＝１とするG_3の入力側の論理条件は，－（ｂ＋（－ｃ））＝１である。この式からa_{12}＝１とする外部入力パターン（＊，０，１）が求まる。従ってa_{12}↑はa_7↓とa_9↓に各々遅延を加えたものの最大のものとなる。またG_3の出力側を０とする入力側の論理条件は，－（ｂ＋（－ｃ））＝０である。その外部入力パターンは（＊，１，＊）または（＊，＊，０）である。従ってa_{12}↑はゲート入力のパターン（１，＊）と（＊，１）の遅延の遅いものとなる。

【００５０】従って，a_{12}↑，a_{12}↓は次のように求まる。
a_{12}↑＝ＭＡＸ（a_7↓＋１０，a_9↓＋１０）＝１８
a_{12}↓＝ＭＡＸ（a_7↑＋７，a_9↑＋７）＝１７
である（但し，ＭＡＸは括弧内の値のうち大きい方を選択することを表す）。

【００５１】a_{13}はa_{12}に$a_{12} - a_{13}$間の配線遅延時間を加算したものである。即ち，
a_{13}↑＝a_{12}↑＋１
a_{13}↓＝a_{12}↓＋１
である。

【００５２】次にa_{14}の遅延時間を計算する。a_{14}の論理値が１となる場合について考える。このような出力となる（a_{11}，a_{13}）の組は（０，０）と（１，１）である。

【００５３】a_{11}，とa_{13}の論理式はそれぞれ（－ａ）・（－ｃ），（－ｂ）・ｃと表すことができる。そこで，（０，０）のパターンを実現する論理式は
（－（（－ａ）・（－ｃ））・（－（（－ｂ）・ｃ））
＝（ａ＋ｃ）・（ｂ＋（－ｃ））
＝（ａ・ｂ）＋ａ・（－ｃ）＋ｂ・ｃ＝ａ・（－ｃ）＋ｂ・ｃ
であるから

a・(－c)＋b・c＝1

とするものである（前述）。

【００５４】この関係から，G_4 の入力が（0，0）となる組合せは（a，b，c）＝（1，＊，0），（＊，1，1）が求まる（＊は0でも1でも良いことを表している）。

【００５５】一方，G_4 の入力が（1，1）となる論理式は

(－a)・(－c)・(－b)・c＝1

である。しかし，(－a)・(－c)・(－b)・cはどのようなa，b，cの組合せに対しても0であるので，(－a)・(－c)・(－b)・c＝1となる組合せはなく，G_4 の入力を（1，1）とする入力（a，b，c）は存在しない。

【００５６】従って，$a_{11}\uparrow$，$a_{13}\uparrow$は考慮する必要がない。a_{14}のダウン遅延は，

$a_{14}\downarrow$＝MAX（$a_{11}\downarrow$＋7，$a_{13}\downarrow$＋7）＝25

と求めるだけで良い。

【００５７】仮に，従来のように入力パターンの組合せを考慮しないと，

$a_{14}\downarrow$＝MAX（MAX（$a_{11}\downarrow$＋7，$a_{13}\downarrow$＋7），MAX（$a_{11}\uparrow$＋7，$a_{13}\uparrow$＋7））＝26

となる。

【００５８】a_{14}＝0となる場合も，同様にして，このような出力値を与えるG_4 の入力値は（0，1），（1，0）であり，このようなパターンを与える入力パターンの組合せは（0，＊，0），（＊，0，1）である。前者のパターンはa_{11}＝0，a_{13}＝1となり，後者のパターンはa_{11}＝1，a_{13}＝0となる。従って，アップ遅延時間は，次のように求まる。

【００５９】$a_{14}\downarrow$＝MAX（MAX（$a_{11}\downarrow$＋9，$a_{13}\uparrow$＋9），MAX（$a_{11}\uparrow$＋9，$a_{13}\downarrow$＋9））＝28である。a_{15}はa_{14}に配線遅延時間を加えたものであり，出力ノードにおけるアップ遅延は29，ダウン遅延は26が得られる。

【００６０】図4は本発明の実施例構成1を示す。図4は論理式もしくは論理情報の記憶により有効なゲート入力パターンを求める場合の構成を示す。図4において，20は制御部であって，各ブロック間のデータの流れを制御するものである。

【００６１】21は入力部であって，回路データを読み込んで回路データ記憶部23に書き込むものである。22は遅延時間演算手段である。

【００６２】23は回路データ記憶部であって，ゲートパラメータ，配線パラメータ，ネットリスト等を保持するものである（図1の回路データ保持手段に相当する）。24は配線／ゲート遅延時間記憶部であって，配線遅延時間，ゲート遅延時間等を保持するものである（図1の配線／素子遅延時間保持部に相当する）。

【００６３】25はパターン／遅延時間記憶部であって，有効な外部入力パターン，ノードの論理式（もしくは真理値表等の論理情報），ノードの遅延時間等を保持するものである（図1のパターン／遅延時間保持部に相当する）。

【００６４】30は配線／ゲート遅延時間計算部であって，回路データ記憶部23に記憶されている回路データ（配線長，配線幅，素子の特性，ゲート幅等）に基づいて，配線遅延時間，ゲード遅延時間を計算するものである。

【００６５】31はパス解析部であって，回路データに基づいて回路のパスを出力側から入力側にたどり，回路解析を行うものである。32は遅延時間演算部である。

【００６６】33は論理解析部であって，論理パターンの解析を行い，有効なゲート入力パターン，外部入力パターンを求めるものである。34は演算部であって，遅延時間を計算するものである。

【００６７】35は最大遅延時間判定部であって，最大遅延時間を求めるものである。36は出力部である。図5は本発明の実施例構成1のフローを示す。

【００６８】Ｓ１　配線／ゲート遅延時間計算部30は配線，ゲートの遅延時間を求め，配線／ゲート遅延時間記憶部24に格納する。

Ｓ２　パス解析部31は遅延時間の演算対象とするノード（対象ノード）を求める。

【００６９】Ｓ３　処理Ａを実行する（処理Ａは図6参照）。図6は処理Ａのフローである。

Ｓ１　遅延時間演算部32において，ノードは外部入力端子，ゲート端子，ゲート入力端子のどれであるかを判定する。外部入力端子（図3のa_1，a_3，a_6）であればＳ２に進む。ゲート出力端子（図3のa_8，a_{10}，a_{12}，a_{14}）であればＳ３に進む。ゲート入力端子（a_2，a_4，a_5，a_7，a_9，s_{11}，a_{13}）であれば，Ｓ６に進む。

【００７０】Ｓ２　外部入力端子であるので，ノードの遅延時間は0である。

Ｓ３　すべてのゲート入力端子の遅延時間は求められているか判定する。求められていればＳ５に進み，求められていなければＳ４に進む。

【００７１】Ｓ４　遅延時間未定の入力端子ノードに対して処理Ａを実行する（Ｓ６以降の処理）。

Ｓ５　ゲート遅延時間のデータからゲート出力端子の遅延時間を計算する（遅延時間の算出の詳細フローは図7で説明する）。

【００７２】Ｓ６　前段の出力端子の遅延時間は求められているか判定する。求められていればＳ８に進み，求められていなければＳ７に進む。

Ｓ７　出力端子ノードに対して処理Ａを実行する（Ｓ３以降の処理）。

【００７３】Ｓ８　出力端子の遅延時間に対して配線遅延時間を加算したものをノードの遅延時間とする。図7

11

は本発明のゲート出力端子の遅延時間の計算のフローである（図６のＳ５の処理）。

【００７４】Ｓ１　ゲート入力のすべてのパターン（ゲート入力パターン）について以下の処理を行う。

Ｓ２　ゲート入力端子の論理値を表す論理集合を求める（論理解析部の処理）。

【００７５】Ｓ３　すべてのゲート入力端子の論理集合についてＡＮＤをとる（論理解析部の処理）。

Ｓ４　得られた論理集合が空集合か判定する（論理解析部の処理）。

【００７６】Ｓ５　遅延時間を計算する（演算部の処理）。

Ｓ６　ゲート入力パターンによる出力端子の論理値を求める。出力端子の論理値がＨ（アップ遅延）であればＳ７に進む。出力端子の論理値がＬ（ダウン遅延）であればＳ９に進む。

【００７７】Ｓ７　出力端子の論理値がＨとなるパターンのこれまで得られた最大（もしくは最小）遅延と比較する（最大遅延時間判定部の処理）。最大でなければＳ１以降の処理を繰り返す。最大であればＳ８に進む。

【００７８】Ｓ８　新しい遅延時間を最大（もしくは最小）遅延時間とし，Ｓ１に戻る。

Ｓ９　出力端子の論理値がＬとなるパターンのこれまで得られた最大（最小）遅延と比較する。最大（もしくは最小）遅延でなければＳ１に戻る。最大（もしくは最小）遅延であればＳ１０に進む。

【００７９】Ｓ１０　新しい遅延時間を最大遅延時間とし，Ｓ１に戻る。Ｓ１において，ゲートの全てについて処理がなされたら，処理を終了する。図８，図９は二分決定グラフ法により最大遅延時間を計算する方法を示す。

【００８０】図８，図９により二分決定グラフ法により最大遅延時間を計算する方法について説明する。a_{15}から入力側にパスを遡る。a_{11}とa_{13}はともに，未処理なので，まずa_{13}を選択する。そして，再び未入力の入力をもつ二入力ゲートG_3にたどりつくので，先にa_7を選択してさらに遡り，Ｂに到達する。Ｂ（＝a_6）は入力ノードであり，そのグラフは図８（ｅ）のように表される。遅延時間は０である。次にa_7のグラフを作成する。a_7のグラフはa_6のグラフにa_6－a_7間の配線遅延を加えたものである。そこで，図８（ｅ）のグラフの葉の各遅延時間に１を加え，図８（ｅ）のグラフを得る。a_7のグラフが得られたので，次はa_9のグラフを作成する。a_9から遡ると，外部入力ノードa_3に達する。ノードa_3のグラフは図８（ｃ）のように表される。さらに，出力側に戻って，a_8のグラフを作る。a_8はインバータの出力であるから，入力ノードのa_5のグラフの理論値を入れ換えたものが，a_8のグラフとなる。また，遅延はそれぞれの遅延にインバータの遅延を加算したものである（図８（ｇ））。

12

【００８１】$a_8\uparrow = a_5\downarrow + 8 = 9$

$a_8\downarrow = a_5\uparrow + 6 = 7$

を求め，図８（ｅ）のグラフを得る。

【００８２】さらに，配線遅延１をアップ遅延，ダウン遅延に加えるとa_9のグラフが得られる（図９（ｈ））。こうして，a_7とa_9のグラフが得られたので，ゲートG_3の入出力論理を考慮してa_{12}のグラフを作成する。作成されるグラフは図９の（ｋ）のようになる。ここで遅延は，

$a_{12}\uparrow = MAX(a_7\downarrow + 10, a_9\downarrow + 10) = 18$

$a_{12}\downarrow = MAX(a_7\uparrow + 7, a_9\uparrow + 7) = 17$

で計算される。a_{13}のグラフはa_{12}のグラフに配線遅延を加えたものである。そこで，図９（ｌ）のグラフを得る。a_{13}が得られたので，次にa_{11}のグラフを求める。a_{11}もa_{13}同様にして計算すると，図９（ｊ）のグラフが得られる。a_{14}はa_{12}とa_{14}のグラフに配線遅延を加えたものである。そして，図９（ｎのグラフが得られる。遅延時間は，

$a_{14}\downarrow = MAX(a_{11}\downarrow + 7, a_{13}\downarrow + 7) = 25$

$a_{14}\uparrow = MAX(MAX(a_{11}\downarrow + 9, a_{13}\uparrow + 9), MAX(a_{11} + 9, a_{13}\downarrow + 9)) = 28$

である。

【００８３】a_{15}はa_{14}のグラフに配線遅延を加えたものである。そして，図９（ｎ）のグラフが得られる。こうして，出力ノードa_{15}におけるアップ遅延は２９，ダウン遅延は２６が得られる。

【００８４】図１０，図１１，図１２は本発明の二分決定グラフ法による場合のアルゴリズムである。図１０(a)はノードのデータ構造の定義である。

【００８５】ｖｅｒｔｅｘはノードであって，ｈｉｇｈはＨの論理値を持つ枝につながるノードを定義し，ｌｏｗはＬの論理値を持つ枝につながるノードを定義する。ｄｅｌａｙ（延時間）は実数であることを定義する。

【００８６】図１０（b)以降，図１１，図１２は，二入力ゲートにおけるグラフ作成アルゴリズム（最大遅延の場合）である。図１１において，５１は最大遅延を求める処理である（例えば，図１４の演算を行うものである（後述））。

【００８７】５２はグラフを求める処理である。５３はメインルーチンである。メインルーチン５３において，処理６０により図１０の処理６２以降の処理を行う。そして，その処理が全てなされると処理６１により図１２の処理６３以降のリダクションの処理を行う。

【００８８】図１２において，５４はリダクションの処理である。５５は最大遅延を選択する処理であって，葉をリダクションするときにそれぞれの保持する最大リダクションのうち最大のものを選択する処理を行う。

【００８９】図１３は本発明のグラフ法による場合のアルゴリズムにおける＜ｏｐ＞の意味の説明図である。＜ｏｐ＞はゲートの演算を表し，ＡＮＤゲートの場合は，

13

Ａ＜ｏｐ＞Ｂは図示の論理を演算する。Ｘは１もしくは０のいずれでも良いことを表す。

【００９０】図１４は本発明のグラフ法による場合のアルゴリズムにおける＜ｄｅｌａｙｏｐ＞の意味の説明図である。＜ｄｅｌａｙｏｐ＞はゲート出力の遅延時間を計算する演算子を表す。図はＡＮＤゲートの場合の演算論理を示す。

【００９１】Ａ・ｖａｌｕｅ，Ｂ・ｖａｌｕｅともに，１の場合は出力は１であるので，Ａ側のｄｅｌａｙとＢ側のｄｅｌａｙの大きい方を遅延とする。Ａ側のｄｅｌａｙはＡ端子のゲートにおけるｄｅｌａｙとその時の出力側のアップ遅延α_A↑の和であり，Ｂ側のｄｅｌａｙはＢ端子のゲートにおけるｄｅｌａｙとその時の出力側のアップ遅延α_B↑の和であるので，その大きい方を選択する。

【００９２】以下同様に，他の入力論理の場合は図示の論理の演算を行う。図１５は本発明のグラフ法のアルゴリズムのフローチャートである。図１５（a)はメインルーチンの処理である（図１１の５３参照）。

【００９３】Ｓ１　グラフＧ１とＧ２に対して処理１（ａｐｐｌｙ－ｓｔｅｐ）を適用する。

Ｓ２　得られたグラフに対して処理２を行う（ｒｅｄｕｃｅ）。

【００９４】図１５（b)は最大遅延を求める処理であって，処理１（ａｐｐｌｙ－ｓｔｅｐ）である。

Ｓ１　新しいノードを作成する。

【００９５】Ｓ２　グラフＧ１とグラフＧ２のルートノード（ｖ１およびｖ２）の値（ｖａｌｕｅ）に対して演算＜ｏｐ＞を行い，新規ノードの値とする。

Ｓ３　値がＸ（図１３，図１４のＸと異なる）でなければ，Ｓ５に進み，ＸであればＳ４に進む。

【００９６】Ｓ４　グラフの作成処理をする（図１６参照）。

Ｓ５　新規ノードのインデックス（ｉｎｄｅｘ）にｎ＋１を代入する。

Ｓ６　グラフＧ１とグラフＧ２のルートノード（ｖ１およびｖ２）に対して演算＜ｄｅｌａｙｏｐ＞を行って，遅延時間を計算する。

【００９７】図１６は本発明のグラフ法における場合のアルゴリズムのフローチャートであって，図１５のＳ４の処理である。

Ｓ１　ｖ１とｖ２を比較する。ｖ１のｉｎｄｅｘ＜ｖ２のｉｎｄｅｘであればＳ２に進み，ｖ１のｉｎｄｅｘ＝ｖ２のｉｎｄｅｘであればＳ３に進み，ｖ１のｉｎｄｅｘ＞ｖ２のｉｎｄｅｘであればＳ４に進む。

【００９８】Ｓ５　ｖ１のインデックス（ｉｎｄｅｘ）を新規ノードのインデックス（ｉｎｄｅｘ）とする。

Ｓ６　ｖ２のインデックスを新規ノードのインデックス（ｉｎｄｅｘ）とする。

【００９９】Ｓ７　ｖｌｏｗ１とｖｈｉｇｈ１をルートとするグラフ処理１（ａｐｐｌｙ－ｓｔｅｐ）を適用し，できたグラフを新規ノードの右枝（ｌｏｗ）におさめる。

Ｓ８　ｖｌｏｗ２とｖｈｉｇｈ２をルートとするグラフ処理１（ａｐｐｌｙ－ｓｔｅｐ）を適用し，できたグラフを新規ノードの左枝（ｈｉｇｈ）におさめる。

【０１００】図１７は本発明のグラフ法のアルゴリズムにおけるリダクションの処理のフローチャートである。

Ｓ１　左右の枝（ｌｏｗとｈｉｇｈ）が同一ノードを指しているノードを削除する。

【０１０１】Ｓ２　グラフの中のノードのうち左右の枝が互いに同じノードを指しているノードの組を検索し，存在しなければ、終了する。

Ｓ３　存在すれば，これらのノードをひとつにまとめる。

【０１０２】Ｓ４　これらのノードが葉（ｔｅｒｍｉｎａｌ　ｎｏｄｅ）ならば，これらのノードの遅延時間のうち最大（最小）のものを新しいノードの遅延時間とする。

【０１０３】

【発明の効果】本発明によれば，実際には生じない無駄なゲート入力のパターンは計算しないので回路の最大遅延時間もしくは最小遅延時間を正確に計算できるとともに，処理が高速化される。また，二分決定グラフ法を利用した場合には，各ノードの入力パターンの解析を高速に行うことができ，正確な遅延時間を高速に算出する。

【図面の簡単な説明】

【図１】本発明の基本構成を示す図である。

【図２】二分決定グラフ法の説明図である。

【図３】回路例を示す図である。

【図４】本発明の実施例構成１を示す図である。

【図５】本発明の実施例構成１のフローを示す図である。

【図６】本発明の処理Ａのフローを示す図である

【図７】本発明の出力端子の遅延時間の計算のフローを示す図である。

【図８】二分決定グラフ法により最大遅延時間を計算する方法（実施例２）を示す図である。

【図９】二分決定グラフ法により最大遅延時間を計算する方法（実施例２）を示す図である。

【図１０】本発明のグラフ法による場合のアルゴリズムを示す図である。

【図１１】本発明のグラフ法による場合のアルゴリズムを示す図である。

【図１２】本発明のグラフ法による場合のアルゴリズムを示す図である。

【図１３】本発明のグラフ法による場合のアルゴリズムの説明図である。

【図１４】本発明のグラフ法による場合のアルゴリズムの説明図である。

15

【図１５】本発明のグラフ法のアルゴリズムのフローチャートを示す図である。

【図１６】本発明のグラフ法による場合のアルゴリズムのフローチャートを示す図である。

【図１７】本発明のグラフ法による場合のアルゴリズムのフローチャートを示す図である。

【図１８】従来の遅延時間算出方法の説明図である。

【０１２２】

【符号の説明】

１：回路遅延時間演算装置

16

１’：入力部

２：配線／素子遅延時間保持部

３：回路データ保持部

４：遅延時間演算部

５：出力部

１０：論理解析部

１１：演算部

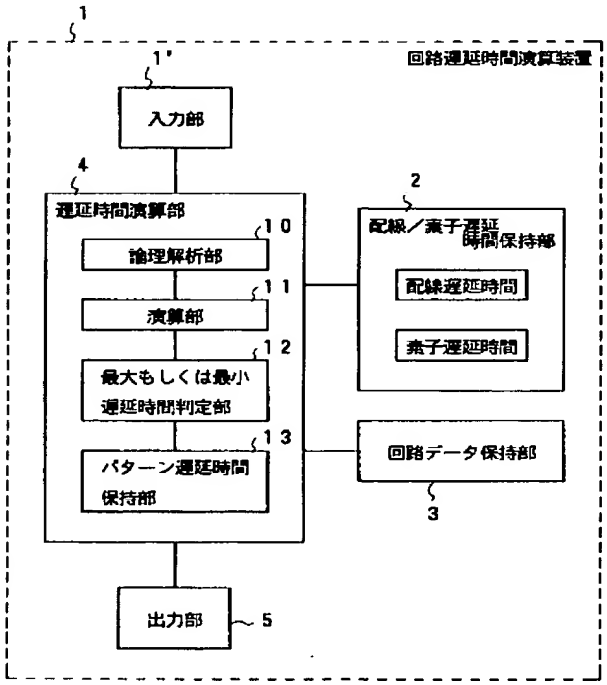
１２：最大もしくは最小遅延時間判定部

１３：パターン遅延時間保持部

10

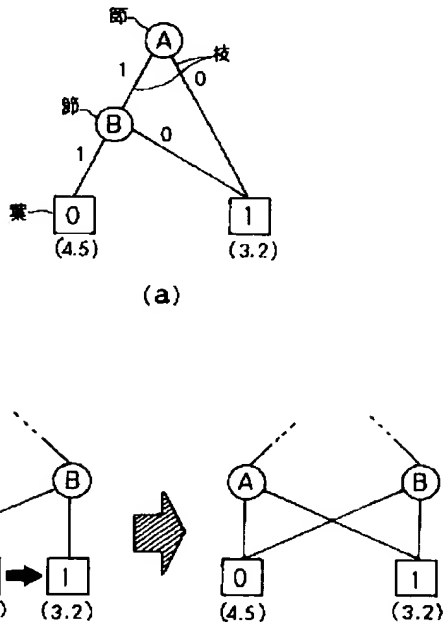
【図１】

本発明の基本構成



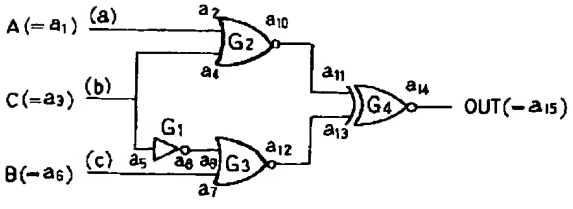
【図２】

二分決定グラフ法の説明図



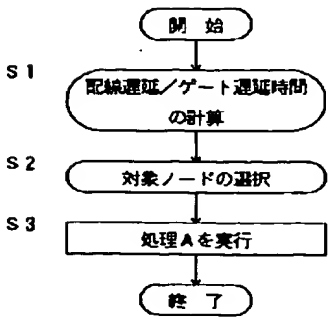
【図３】

回　路　例



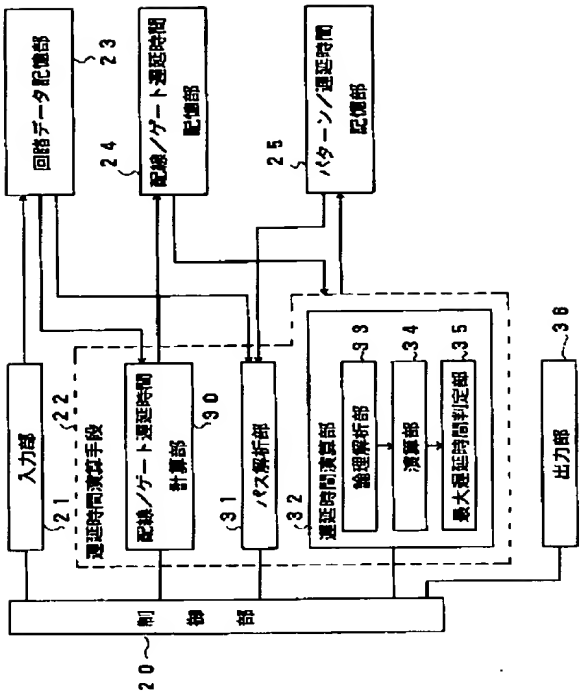
【図５】

本発明の実施例構成1のフロー



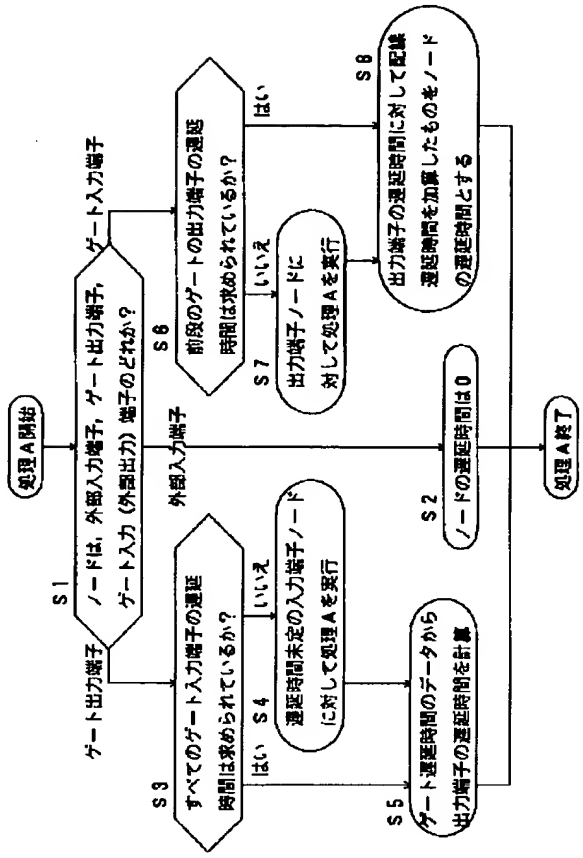
【図４】

本発明の実施例構成１



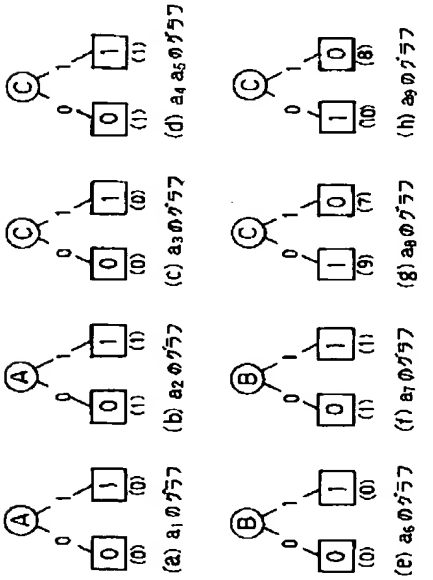
【図６】

処理Ａのフロー



【図８】

二分決定グラフ法により最大遅延時間を計算する方法（実施例2）



【図１３】

本発明のグラフ法による場合のアルゴリズムの説明図

(op) について

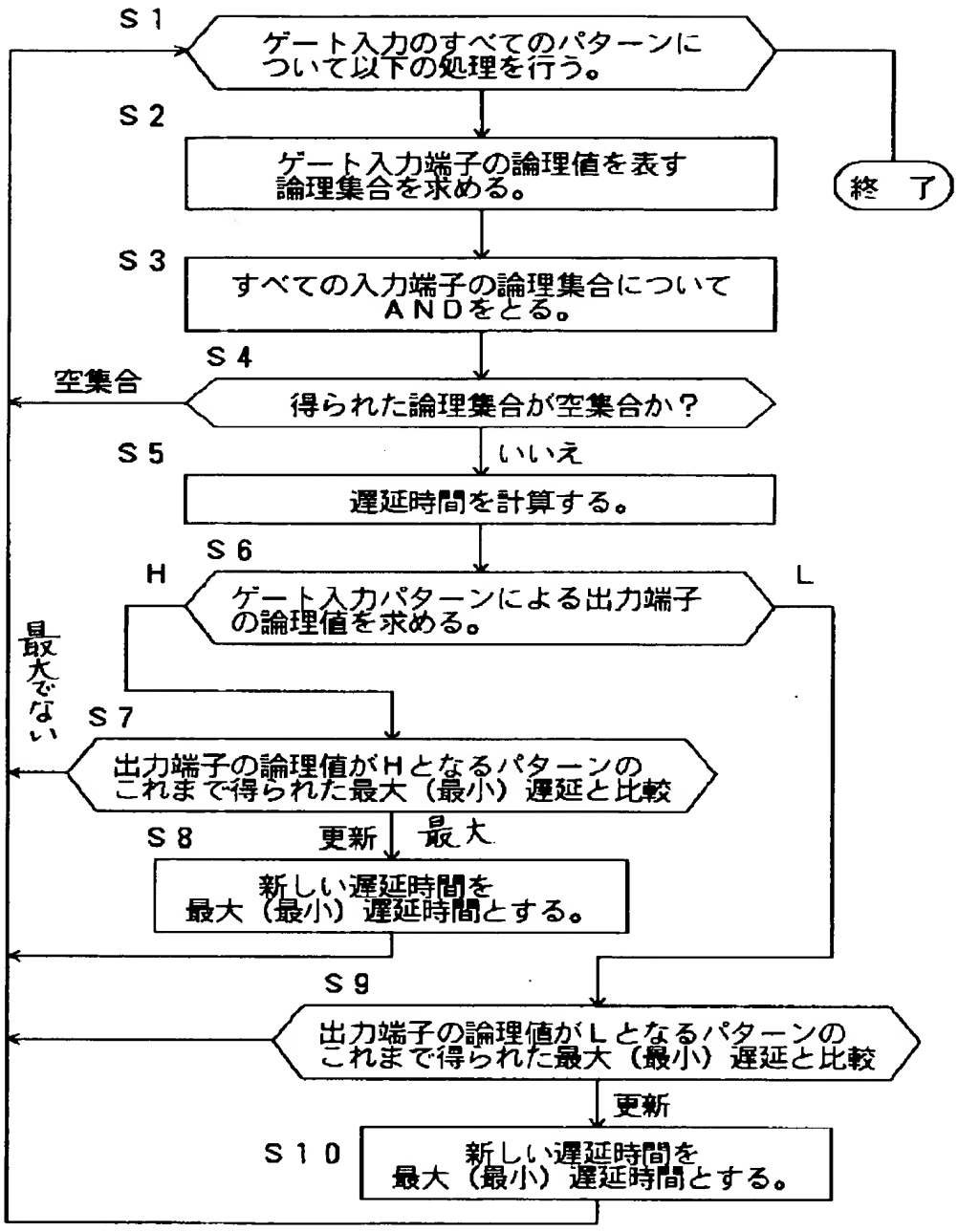
(op) はゲートの演算を表す

(例)　ANDゲート　　A, B : (1, 0, X)

A	B	A (op) B
xxxooo	xoxoxo	xoxooo

【図７】

出力端子の遅延時間の計算のフロー



【図９】

二分決定グラフ法により最大遅延時間を計算する方法（実施例２）

【図１４】

本発明のグラフ法による場合のアルゴリズムの説明図

(delayop) について

(delayop) はゲート出力の遅延時間を計算する演算子を表す

(例)　ANDゲート　　A, B : vertex

A.value	B.value	A (delayop) B
	1	MAX(A.delay + $\alpha_{A\uparrow}$, B.delay + $\alpha_{B\uparrow}$)
	0	B.delay + $\alpha_{B\downarrow}$
	x	— (不定)
1	1	MAX(A.delay + $\alpha_{A\uparrow}$, B.delay + $\alpha_{B\uparrow}$)
1	0	A.delay + $\alpha_{A\downarrow}$
1	x	— (不定)
0	1	MIN(A.delay + $\alpha_{A\downarrow}$, B.delay + $\alpha_{B\uparrow}$)
0	0	A.delay + $\alpha_{A\downarrow}$
0	x	— (不定)
x	1	B.delay + $\alpha_{B\uparrow}$
x	0	B.delay + $\alpha_{B\downarrow}$
x	x	— (不定)

$\alpha_{A\uparrow}$: Aに１の入力があった時，出力が１となるまでの遅延

$\alpha_{A\downarrow}$: Aに０の入力があった時，出力が０となるまでの遅延

【図１０】

本発明のグラフ法による場合のアルゴリズム

(a)　ノードのデータ構造の定義

```
type vertex = record
        low, high : vertex;
        index: 1..n+1;
        value:(0,1,X);
        delay:real number
        id:integer;
        mark:boolean;
    end;
```

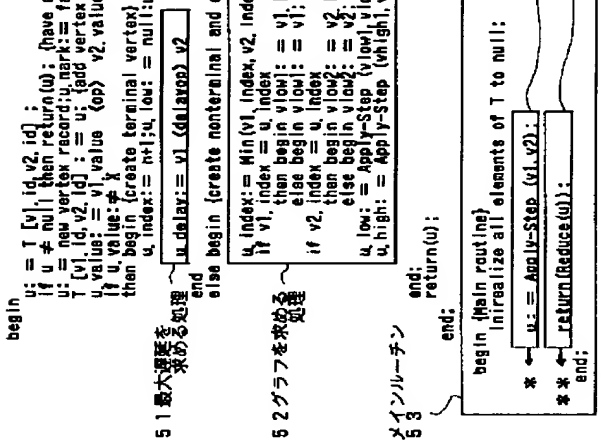
(b) ニ入力ゲートにおけるグラフ作成アルゴリズム（最大遅延の場合）

```
function Apply(v1,v2: vertex; (op) : operator; (delayop) : operator):vertex;
    var T: array [1 .. |G1| , 1 .. |G2| ] of vertex;
        {Recuraive routine to implement Apply}
*   function Apply-Step (v1,v2: vertex): vertex;
```

S 2

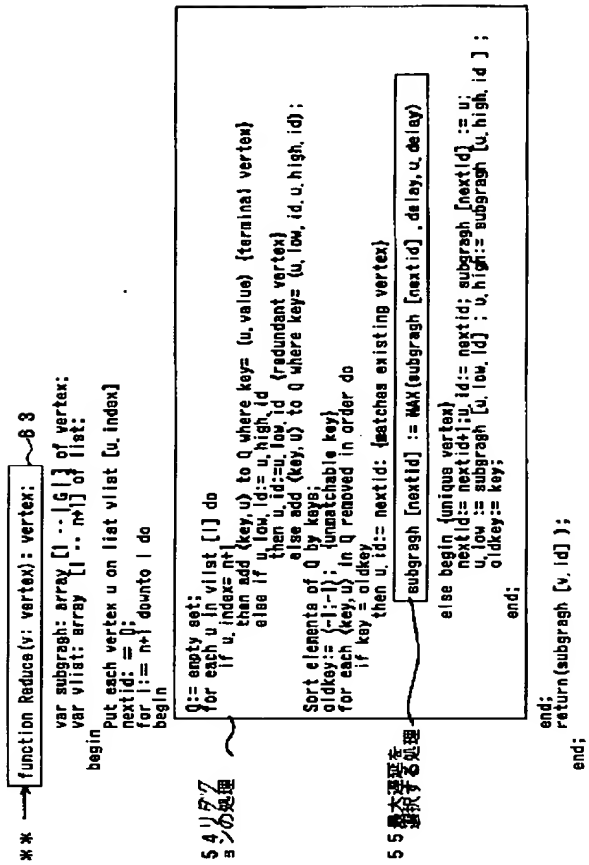
【図１１】

本発明のグラフ法による場合のアルゴリズム



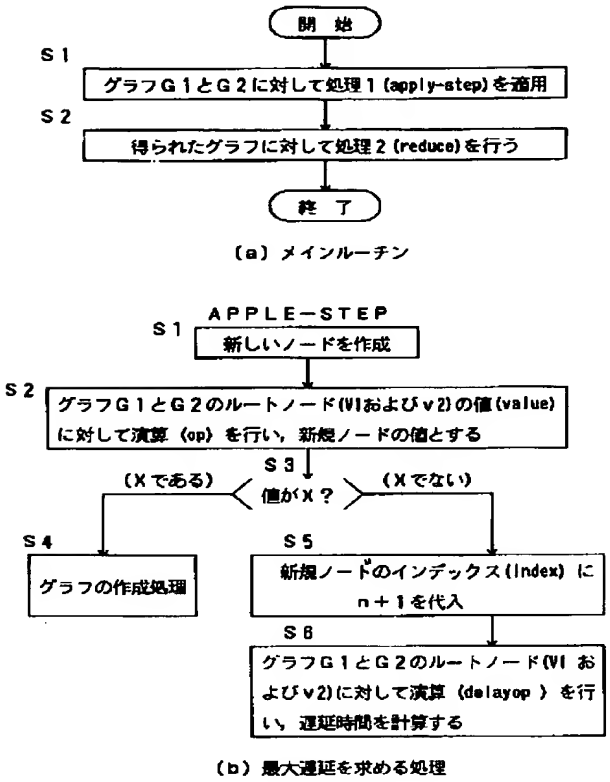
【図１２】

本発明のグラフ法による場合のアルゴリズム



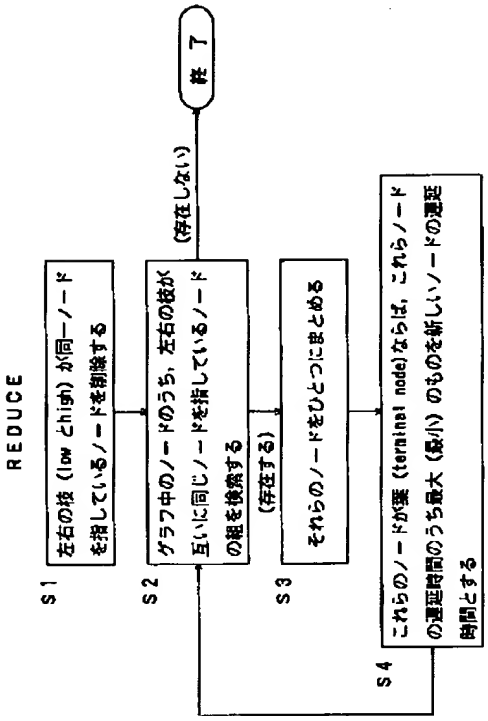
【図１５】

本発明のグラフ法のアルゴリズムのフローチャート



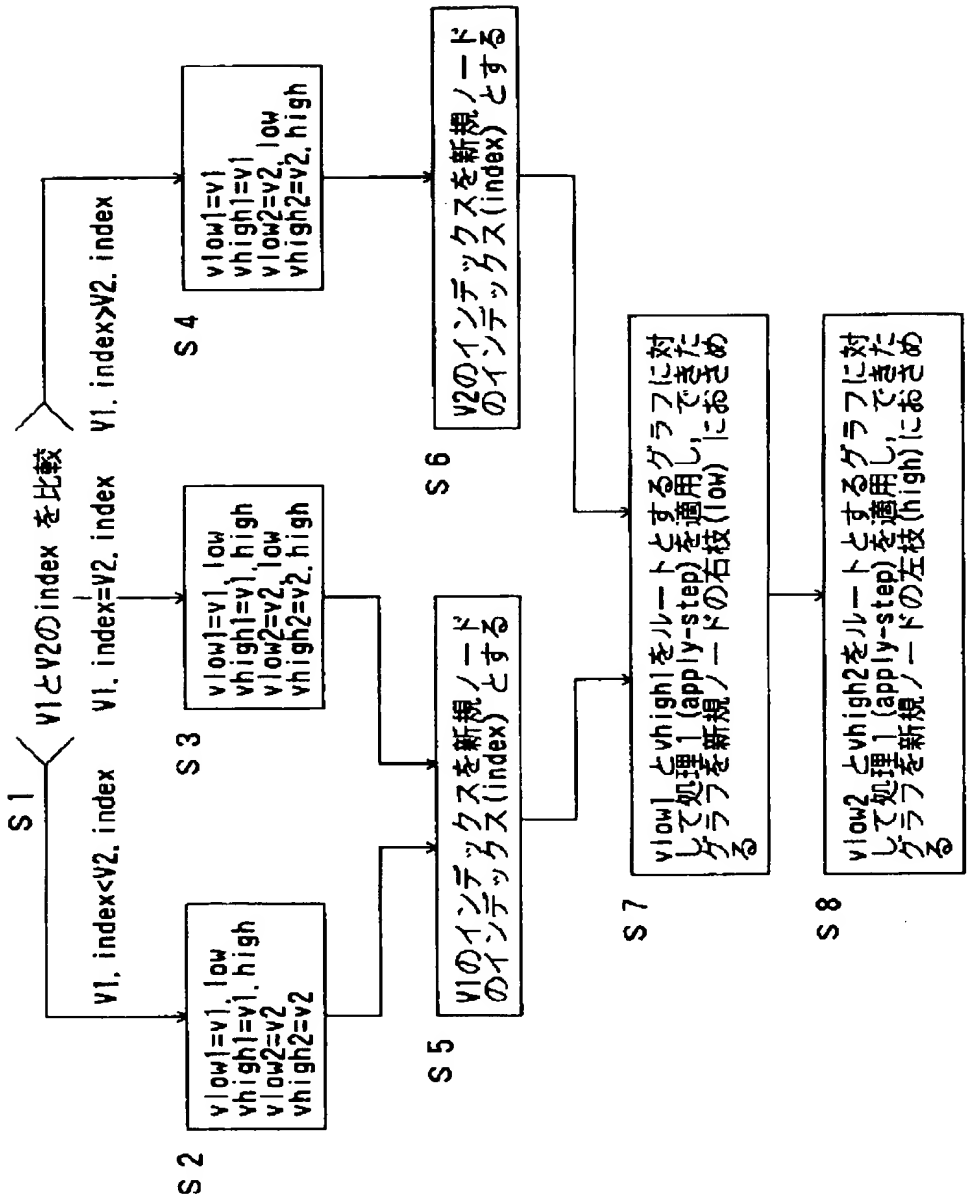
【図１７】

本発明のグラフ法による場合のアルゴリズムのフローチャート



【図１６】

本発明のグラフ法による場合のアルゴリズムのフローチャート



【図１８】

従来の遅延時間算出方法の説明図